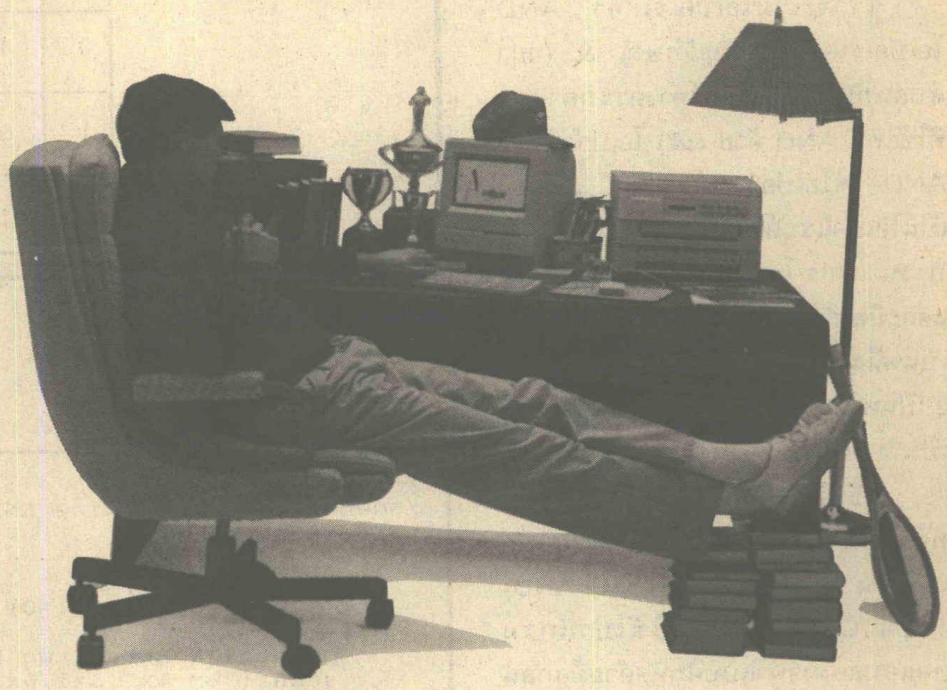
ลออ เพิ่มสมบัติ

การเขียน กราฟิก อย่างมีระบบ ตอน การเข้าถึงหน่วยความจำโดยตรง

ทำความเข้าใจกับเทคนิคการเข้าถึงหน่วยความจำโดยตรง ซึ่งเป็นส่วนสำคัญในการทำงานในโหมดกราฟิก



การทำความเข้าใจเทคนิคการเข้าถึงหน่วยความจำโดยตรง (Direct Memory Access) หรือเรียกสั้น ๆ ว่า DMA เป็นหัวใจสำคัญในการเขียนรูปต่าง ๆ ในโหมดกราฟิกสี

การจะใช้ประโยชน์สูงสุดของ DMA จะต้องเข้าถึง C bitwise operator ซึ่งอนุญาตให้บิตแต่ละบิตในหน่วยความจำถูกเข้าถึงและถูกจัดการได้ ดังนั้นเราจึงเริ่มศึกษาโอเปอเรเตอร์บิตไวซ์ หลังจากนั้นก็จะศึกษามโนภาพของการแสดงผลแผนที่ความจำ แล้วจะแสดงวิธีที่ DMA ถูกใช้สำหรับโปรแกรมเวิร์ดโปรเซสซิ่ง ซึ่งจำเป็นต้องใช้ far pointers (เป็นพอยน์เตอร์ที่ใช้ไปได้ทุกหนทุกแห่งในหน่วยความจำ) พื้นฐานสำคัญในการที่จะศึกษาเรื่องนี้คือเลขฐาน 16 กับเลขฐาน 2

ชนิดของข้อมูลคงที่ อาทิเช่น ตัวอักษร จำนวนเต็มทศนิยม และอื่น ๆ เราไม่เคยได้ศึกษาถึงสิ่งที่อยู่เบื้องหลังชนิดข้อมูลเหล่านี้เลยว่ามันถูกสร้างขึ้นมาได้อย่างไร แท้จริงแล้วมันเกิดจากแต่ละบิต การจัดการบิต ซึ่งมีความสำคัญมากในการเขียนโปรแกรม โดยเฉพาะอย่างยิ่งถ้าต้องการติดต่อกับฮาร์ดแวร์โดยตรง ถ้าจะศึกษาโปรแกรมเน้นที่ข้อมูลฮาร์ดแวร์ก็ต้องเน้นที่บิต เครื่องมือฮาร์ดแวร์ต้องการข้อมูลนำเข้าและข้อมูลนำออกในรูปของบิตมากกว่าในรูปของไบต์ (ที่ออกมาเป็นตัวอักษรหรือจำนวน)

C จะใช้โอเปอเรเตอร์ Bitwise อยู่ 6 ตัว ซึ่งพอสรุปได้ดังนี้

โอเปอเรเตอร์	สัญลักษณ์
AND	&
Inclusive OR	:
Exclusive OR (XOR)	-
Right shift	>>
Left shift	<<
Complement	~

ซึ่งโอเปอเรเตอร์เหล่านี้จะใช้ได้กับข้อมูลชนิดตัวอักษรและจำนวนเต็ม (ทั้งแบบมีเครื่องหมายและไม่มีเครื่องหมาย) แต่ใช้กับจำนวนทศนิยมไม่ได้ เนื่องจากโอเปอเรเตอร์มีผลต่อบิต

บิตจะเริ่มจากขวาไปซ้ายดังรูปที่ 1

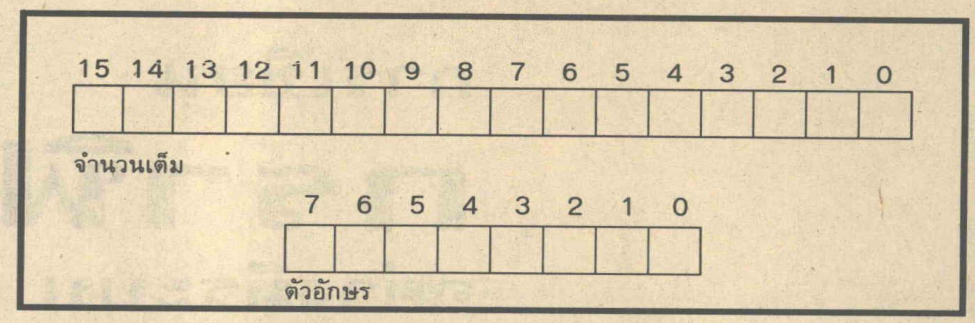
เราจะเริ่มโอเปอเรเตอร์ AND โอเปอเรเตอร์นี้ใช้สัญลักษณ์ & (อย่าสับสนกับโอเปอเรเตอร์ทางตรรกศาสตร์ที่ใช้แทน AND ด้วย &&) โอเปอเรเตอร์ AND ใช้โอเปอแรนด์ 2 ตัว และต้องเป็นโอเปอแรนด์ชนิดเดียวกัน แนวคิดคือการนำเอา 2 โอเปอแรนด์มาเทียบกับบิตแต่ละบิต ถ้าบิตที่ 0 ของโอเปอแรนด์แรกเป็นหนึ่งและบิตที่ 0 ของโอเปอแรนด์ที่ 2 เป็นหนึ่งเมื่อนำเอามา and กัน ผลลัพธ์เป็น 1 ซึ่งจะสรุปได้ดังรูปที่ 2

วิธีที่ดีที่สุดคือการเขียนโปรแกรมทดสอบการใช้โอเปอเรเตอร์ AND กับเลขฐาน 16 ด้วยโปรแกรม AND-TEST.C แต่ก่อนที่จะใช้โปรแกรมทดสอบลองมาทำแบบฝึกหัดด้วยมือก่อน เช่น ต้องการจะนำเอา C ในเลขฐาน 16 มา AND กับ 7 ในเลขฐาน 16

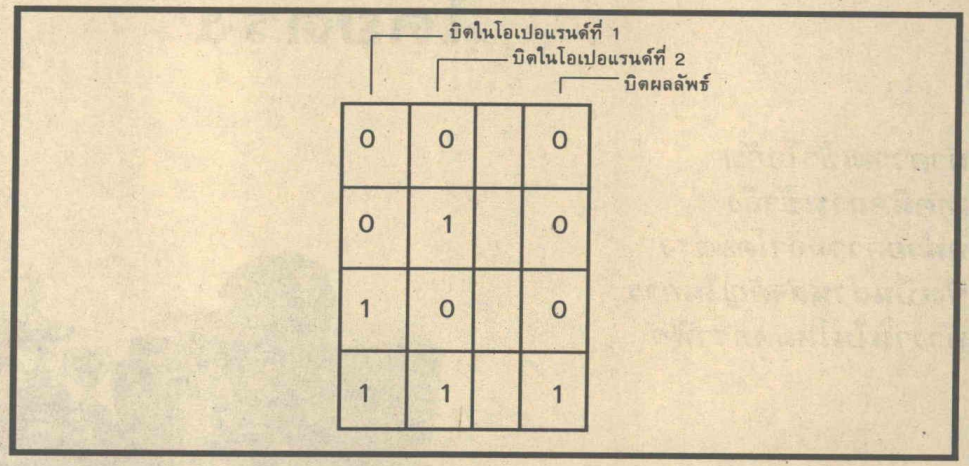
เนื่องจากเลขฐาน 16 ประกอบด้วย 0-15

โดยใช้ A แทน 10 ในเลขฐาน 10

B แทน 11
C แทน 12
D แทน 13
E แทน 14
F แทน 15



รูปที่ 1 การเรียงเริ่มจากขวามาซ้าย



รูปที่ 2

โปรแกรม andtest.c

```
/*andtest.c*/
/*demonstrates bitwise AND operator*/
void main(void)
{
  unsigned int x1,x2;
  while(x1 !=0 || x2 !=0)  /*terminate on 0,0 */
    {
     printf("\nEnter two hex numbers (ff or less):");
     scanf("%x %x",&x1,&x2);
     printf("%02x & %02x=%02x\n",x1,x2,x1&x2);
     }
    }
```

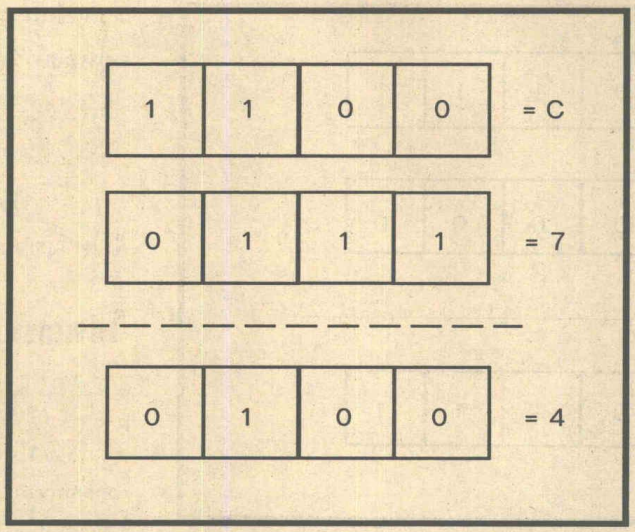
ถ้าเปลี่ยน C = 12 ให้เป็นเลขฐาน 2 ก็ลองเอา 2 หารแล้วเอาเศษมาตอบโดยไล่จากล่างขึ้นบน ก็จบด้วยฐาน 2 ตามต้องการ จะได้ว่า 12 ในเลขฐาน 16 เป็น 1100 ในเลขฐาน 2

ดังนั้น 7 เขียนเป็นฐาน 2 ได้ 111

ดังนั้น OC&07 = 04

นั่นคือรูปที่ 3

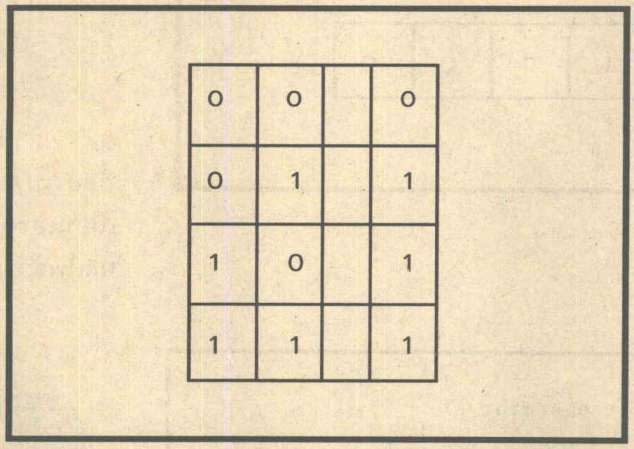
รูปที่ 3
แสดงการใช้ AND (2)



โปรแกรม ortest.c

```
/*ortest.c*/
/*demonstrates bitwise OR operator*/
void main(void)
{
  unsigned int x1,x2;
  while(x1 !=0|| x2 !=0) /*terminate on 0,0*/
    {
    printf("\Enter two numbers (ff or less): ");
    scanf("%x %x" ,&x1,&x2);
    printf("%02x | %02x=%02x\n",x1,x2,x1 | x2);
    }
  }
```

รูปที่ 4
ผลที่ได้จาก OR (:)



บิตไวซ์ OR (:)

โอเปอเรเตอร์ Bitwise ที่สำคัญอีกอันหนึ่งคือ OR ที่แทนด้วยสัญลักษณ์ : เมื่อ 2 บิตมาทำกันด้วย OR ผลลัพธ์จะเป็น 1 ถ้ามีโอเปอแรนด์เป็น 1 อย่างน้อย 1 ตัว ถ้าไม่มีบิตใดเป็น 1 เลย ผลลัพธ์จะเป็น 0

ลองใช้โปรแกรม ortest.c ทดสอบการใช้ OR ผลลัพธ์จากโปรแกรมคือ

```
00 : 01 = 01
01 : 00 = 01
01 : 01 = 01
0c : 07 = 0f
ad : cb = ef
```

ถ้าคุณยังคลุมเครือหรือไม่กระจ่างชัดกับผลลัพธ์ที่ได้ทั้งในโปรแกรม ortest.c และ andtest.c คุณจะต้องพิสูจน์หรือหาให้ได้โดยการเปลี่ยนเลขฐาน 16 ให้เป็นเลขฐานสอง แล้วลองใช้โอเปอเรชัน OR ทีละคู่ แล้วค่อยเปลี่ยนผลลัพธ์ที่ได้ให้เป็นเลขฐาน 16 อีกทีหนึ่ง

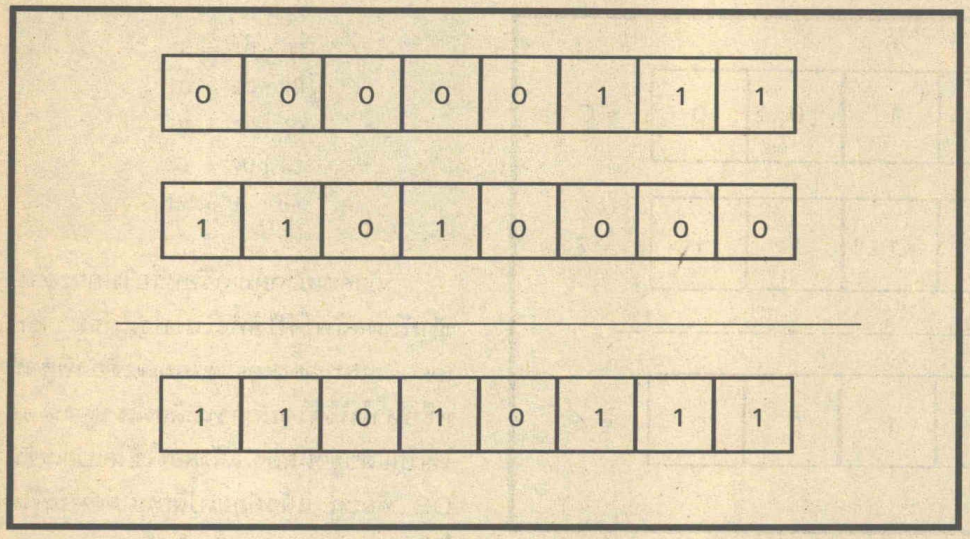
บิตไวซ์โอเปอเรเตอร์ OR นำมาใช้กันบ่อย ๆ ในการเชื่อมบิตจากตัวแปรที่ต่าง ๆ กันให้เป็นตัวแปรเดี่ยว ตัวอย่างเช่น สมมติว่ามีตัวแปรตัวอักษร 2 ตัว คือ ch1 กับ ch2 และบิต 0 ถึง 3 ของ ch1 บรรจุค่าที่เราต้องการ ขณะที่บิต 4 ถึง 7 ของ ch2 ก็เป็นค่าที่เราต้องการด้วย เราจะกำหนดให้ส่วนที่เราไม่ต้องการทั้งสองตัวแปรเป็น 0 ทั้งหมด นั่นคือเราจะเชื่อมตัวแปร 2 ตัวด้วย : แล้วไปเก็บไว้ที่ ans

```
ans = ch1 : ch2
```

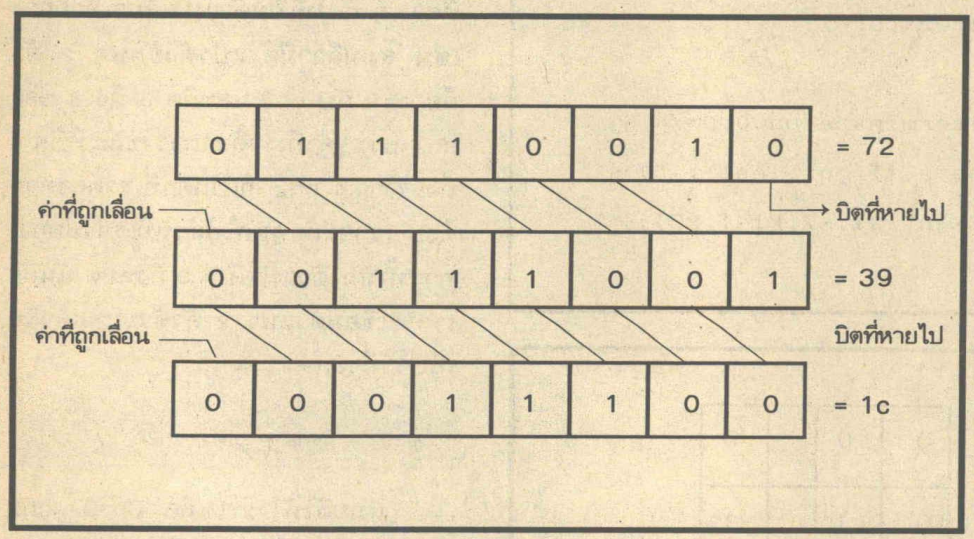
สมมติให้ ch1 คือ 0×07 ch2 คือ 0×d0 จะได้ผลลัพธ์ดังรูปที่ 5

โอเปอเรเตอร์บิตไวซ์เลื่อนขวา (Bitwise Right - Shift >>)

นอกจากโอเปอเรเตอร์บิตไวซ์ที่ทำกับ 2 ตัวแปร ยังมีโอเปอเรเตอร์บิตไวซ์ที่กระทำกับตัวแปรตัวเดียว อาทิ โอเปอเรเตอร์บิตไวซ์เลื่อนขวา โดยใช้สัญลักษณ์เครื่องหมายมากกว่าสองตัวติดกัน (>>) โอเปอเรเตอร์ตัวนี้จะทำการเลื่อนแต่ละบิตในโอเปอแรนด์ไปทาง



รูปที่ 5 ผลลัพธ์ที่ได้จาก OR (:)



รูปที่ 6 แสดงการเลื่อนขวา

โปรแกรม shiftest.c

```
    /*shiftest.c*/
/* demonstrates bitwise right shift operator */
void main(void)
 {
   unsigned int x1,x2;

   while (x1 !=0)     /*terminate on x1=0*/
     {
     printf("\nEnter hex number (ff or less and number of bits");
     printf("\n to shift(8 or less;example 'cc 3'):");
     scanf("%x %d ",&x1,&x2);
     printf("%02x >> %d=%02x\n",x1,x2,x1>>x2);
     }
   }
```

ขวา จำนวนที่จะเลื่อนขึ้นอยู่กับจำนวนที่อยู่หลังเครื่องหมาย >> เช่น

```
ch >> 3
```

ทำให้ทุกบิตใน ch ถูกเลื่อนขึ้นไปทางขวา 3 ตำแหน่ง ดังรูปที่ 6

โปรแกรม shifttest.c

```
/* shifttest.c */
/* demonstrates bit wise right shift operator */
```

ถ้าเราป้อน 80 1

```
80 >> 1 = 40
```

ถ้าเราป้อน 80 2

```
80 >> 2 = 20
```

ถ้าเราป้อน 80 7

```
80 >> 7 = 01
```

ถ้าเราป้อน fo 4

```
fo >> 4 = of
```

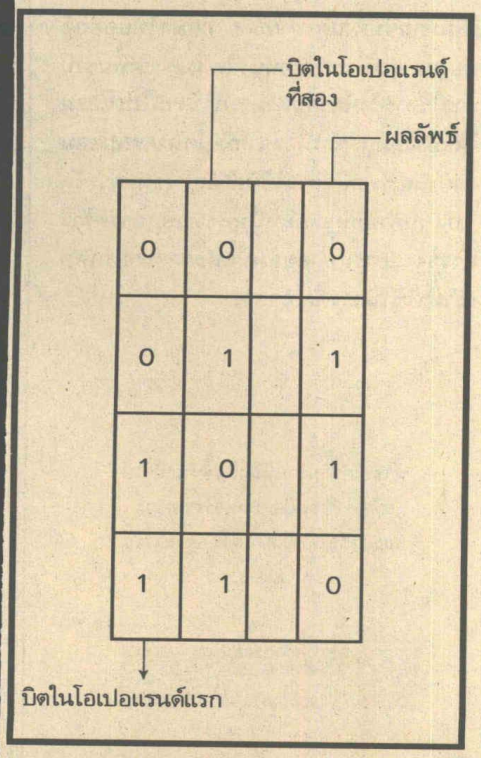
ถ้าผลลัพธ์ที่ได้ยังไม่เข้าใจแจ่มชัด ก็ลองเปลี่ยนให้เป็นเลขฐาน 2 แล้วลองเลื่อนเองด้วยมือ พร้อมกับตรวจสอบผลลัพธ์ว่าเหมือนกับโปรแกรมหรือไม่

โอเปอเรเตอร์เกี่ยวกับตรรกตัวอื่น

มีโอเปอเรเตอร์บิตไวซ์ทั้งหมด 6 ตัวด้วยกัน เราได้กล่าวมาแล้ว 3 ตัว อีก 3 ตัวเราจะพูดถึงต่อไป และจะแสดงให้เห็นถึงการคำนวณเกี่ยวกับบิตไวซ์ทั้ง 6 ตัวด้วยโปรแกรม bitcalc.c คุณก็ลองคีย์โปรแกรมคอมพายล์ แล้วทดลองรันพร้อมกับศึกษาผลลัพธ์ก็จะทำให้คุณเข้าใจขึ้น

บิตไวซ์ XOR (^)

บิตไวซ์ OR (:) ที่ได้กล่าวไว้แล้วจะให้ค่า 1 เมื่อบิตใดบิตหนึ่งเป็น 1 หรือทั้งคู่ บิตไวซ์เอ็กซ์คลูซีฟ OR หรือ XOR จะให้ค่า 1 เมื่อบิตใดบิตหนึ่งเป็น 1 แต่ไม่ใช่ทั้งคู่ XOR จะแทนด้วยเครื่องหมาย caret (^) ดูจากรูปที่ 7



รูปที่ 7 แสดงผลลัพธ์ที่เกิดจาก XOR

โปรแกรม bitcalc.c

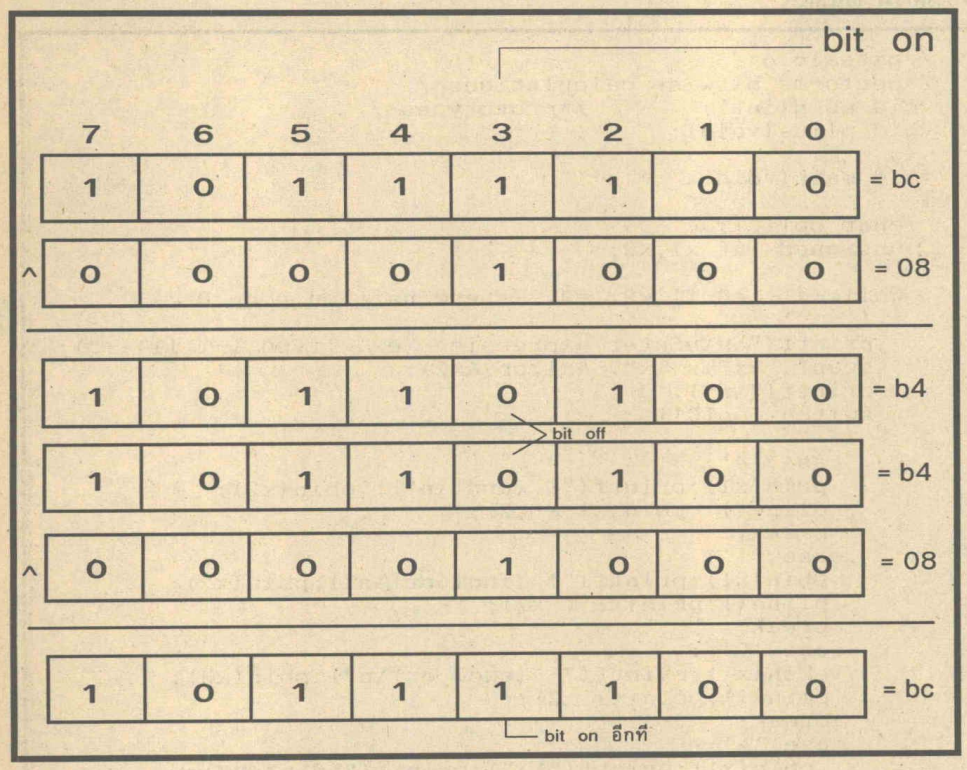
```
/*bitcalc.c*/
/*performs bitwise calculations*/
void pbin(int);        /*prototypes*/
void pline(void);

void main(void)
{
  char op[10];
  unsigned int x1,x2;

  while(x1 !=0 || x2 !=0) /*terminate on 0 op 0*/
    {
    printf("\n\nEnter expression (ex. 'ff00 & 1111'):");
    scanf("%x %s %x", &x1,op,&x2);
    printf("\n");
    switch (op[0])
     {
      case '&':
       pbin(x1);printf("& (and)\n"); pbin(x2);
       pline(); pbin(x1 & x2);
       break;
      case '|':
       pbin(x1);printf("| (incl or)\n");pbin(x2);
       pline();pbin(x1 | x2);
       break;
      case '^' :
       pbin(x1);printf("^ (excl or)\n");pbin(x2);
       pline();pbin(x1^x2);
       break;
      case '>':
        pbin(x1);printf(">>");printf("%d\n",x2);
        pline();pbin(x1>>x2);
        break;
      case '<':
       pbin(x1);printf("<<"); printf("%d\n",x2);
       pline();pbin(x1<<x2);
       break;
      case '~':
       pbin(x1);printf("~ (complement)\n");
       pline();pbin(~ x1);
       break;
     default: printf("Not valid operator.\n");
     }
   }
  }

  /*pbin*/
  /*prints number in hex and binary*/
  void pbin(int num)
  {
  unsigned int mask;
  int j,bit;
  mask =0x8000;                    /*one bit mask*/
  printf("%04x ",num);             /*print in hex*/
  for(j=0;j<16;j++)                /*for each bit in num*/
    {
     bit=(mask & num) ? 1:0;       /*bit is 1 or 0*/
     printf("%d",bit);             /*print bit */
     if(j==7)                      /* print dash between*/
       printf("--");               /*bytes */
     mask >>=1;                    /*shift mask to right*/
     }
     printf("\n");
    }

    /*pline()*/
    void pline(void)
    {
    printf("-----------------------------------\n");
    }
```



รูปที่ 8

XOR มีประโยชน์ในการสลับกันไปมาระหว่าง 0 กับ 1

เพราะว่า 1 XOR 1 ได้ 0

1 XOR 0 ได้ 1

เช่น ต้องการจะสลับบิต 3 ในตัวแปร ch ที่เป็นชนิด char ก็ทำได้ดังนี้

```
ch = ch  0 × 08
```

จากรูปที่ 8 แสดงวิธีการสลับไปมาเฉพาะบิต 3 โดยที่บิตอื่นยังคงเดิม สมมติตัวแปร ch เริ่มด้วยค่า 0 × BC

โอเปอเรเตอร์บิตไวซ์เลื่อนซ้าย (<<)

โอเปอเรเตอร์บิตไวซ์เลื่อนซ้าย ก็คือการเลื่อนบิตไปทางซ้ายแทนที่จะไปทางขวา ค่าบิตที่จะใส่ทางขวาต้องเป็น 0 และไม่สนใจเครื่องหมาย (sign) และชนิดของโอเปอแรนด์

โอเปอเรเตอร์บิตไวซ์คอมพลิเมนต์ (~)

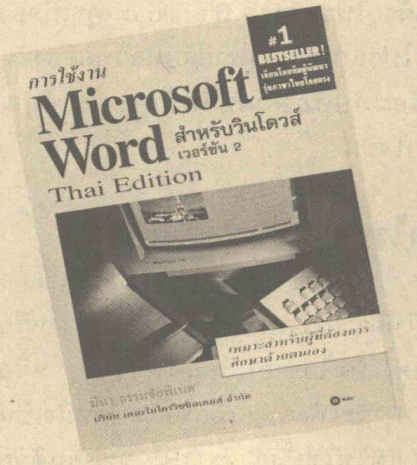
โอเปอเรเตอร์บิตไวซ์คอมพลิเมนต์ (~) กระทำบนโอเปอแรนด์ตัวเดียว โดยเปลี่ยนบิตในโอเปอแรนด์จาก 1 เป็น 0 และจาก 0 เป็น 1 เช่น

```
~03 = = fc
~ffff = = 0
~cc = = 33
~88 = = 77
```

สำหรับในฉบับนี้ขอจบเพียงเท่านี้ ก่อนยังมีเรื่องน่ารู้อีกมาก เอาไว้พบกันในฉบับหน้าครับ

ตรงนี้

สำหรับผู้ที่
ต้องการศึกษา
ด้วยตนเอง



หนังสือเล่มนี้เขียนขึ้นเพื่อตอบสนองความต้องการของผู้ใช้โปรแกรม Word ที่เพิ่มมากขึ้นตั้งแต่ Word ภาษาไทยออกสู่ท้องตลาด เนื้อหาของหนังสือเริ่มตั้งแต่พื้นฐานการใช้งานวินโดว์จนสามารถใช้งานโปรแกรมได้อย่างสมบูรณ์ เนื่องจากมีเนื้อหาครอบคลุมคำสั่งและเทคนิคการใช้ที่สำคัญ เช่น การจัดรูปแบบตัวอักษรและหน้ากระดาษ การสร้างตาราง การแบ่งคอลัมน์และการนำกราฟิกเข้ามาใช้ในการจัดเอกสาร

หาซื้อได้ตามร้านหนังสือชั้นนำทั่วไป
หรือที่ ซีเอ็ดบุ๊คเซ็นเตอร์ทุกสาขา
และสั่งซื้อได้ที่ซีเอ็ด สำนักงานใหญ่
โทร 248-6280

มีจำหน่ายแล้ว

ซีเอ็ด